Semestrálne zadanie: Komunikácia s využitím UDP protokolu

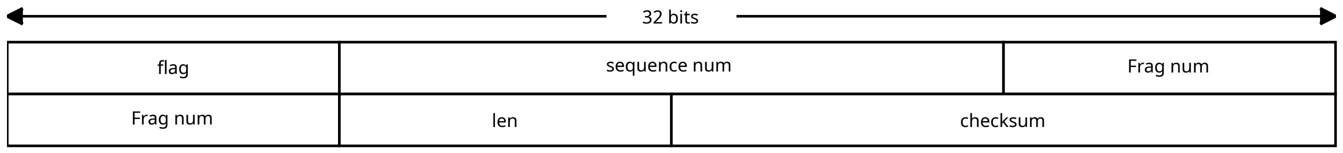
by
Tomáš Meravý Murárik
at Faculty of Informatics and Information Technologies STU

Course: Computer and Communication Networks

Assignment object: Design and implement P2P(Peer to Peer) application using custom protocol built on top of UDP (User Datagram Protocol) in the transport layer of TCP/IP model. The application should allow 2 users to communication over local Ethernet network, including text transmission and exchange of files between computers (nodes) . Both nodes will work simultaneously as receiver and sender.

Table of Contents

Structure of protocol header

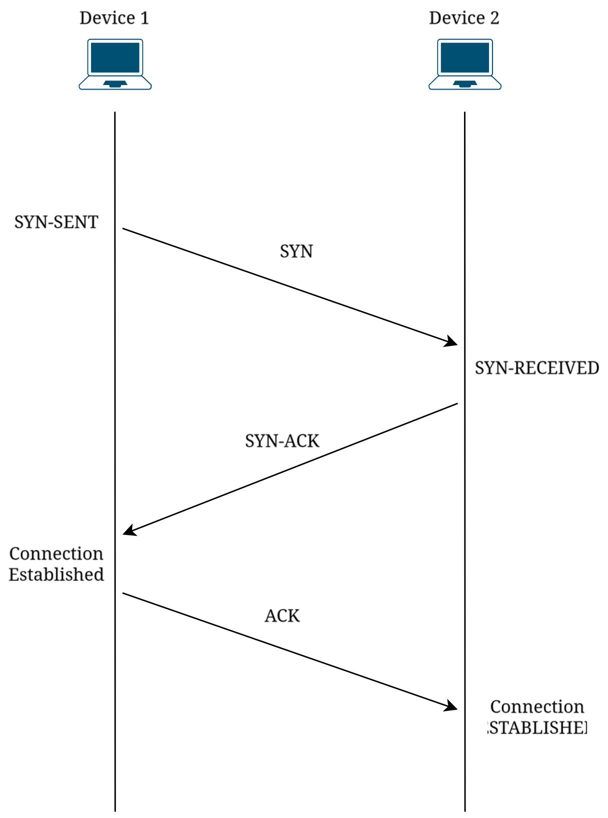


Estabilishing connection:

Similaraly to TCP I will be using 3 way handshake using SYN-ACK system.

> SYN) Both clients will be sending SYN packets to specified ports till one of them responds with SYN-ACK.
>
> SYN-ACK)After SYN packet is recievied they will send back SYN-ACK packet acknowledging that they received SYN packet and is waiting for his ACK.
>
> ACK) After client gets SYN-ACK packet he will respond with ACK packet completing the 3 way handshake.

Protocol uses the Flags field to signal which control state it’s using.



Fragmentation:

Based on sequence number and fragment number , the program will determine how many packets it should expect and how it will reassemble the packet back together once all packets have been received.

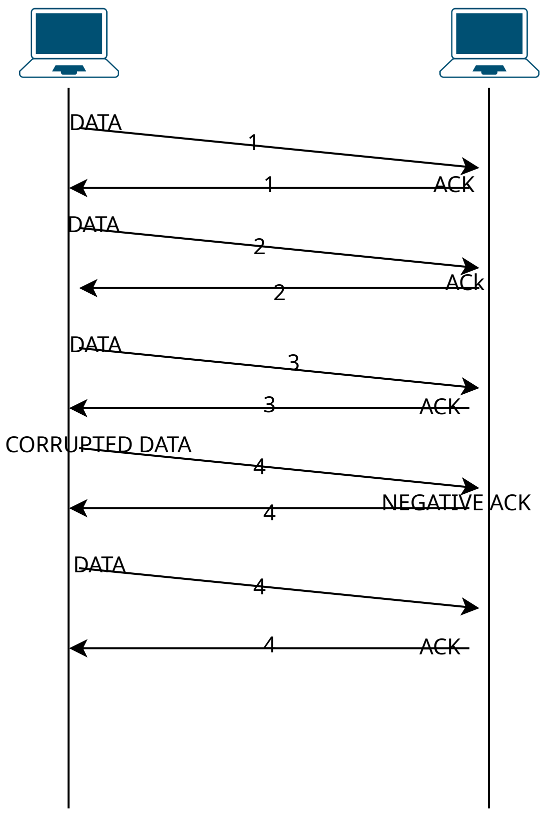
Error detection :

Using CRC16 the program will use checksum value to determine whether the received packet is corrupted or not.

CRC16:this algorithm uses 16 bit polynomial (divisor) to perform bitwise division on the data using   binary xor operations where the remainder of there operation is appended at the end of the header . Same operation is performed by the receiver and if the receivers operation is equal to the checksum he will return True saying that the data is intact . If the operation is not equal the algorithm will return False and the receiver will ask for the data again.

Reliable Data Transmission (ARQ):

After receiving each packet, the receiver sends an acknowledgment packet confirming successful reception. If the packet is corrupted, a negative acknowledgment (NACK) is sent, requesting retransmission. If a fragment is missing during larger data transfers, the receiver will ask for it using its sequence number, identifying the missing fragment.
Using Stop-and-Wait ARQ, the sender will wait for an acknowledgment (ACK) before sending the next packet. If the sender receives a NACK or times out waiting for an ACK, it will resend the packet until an ACK is received.



Keep Alive:

The keep-alive mechanism will periodically send messages (e.g., every 30 seconds) from the sender to the receiver to indicate that the connection should remain active. The receiver is expected to respond with an acknowledgment (ACK). If no ACK is received after several keep-alive attempts, the connection will be closed.

Data corruption simulation :

To check whether error detection and fragmentation works , the sender will have option to send bad packet on purpose by altering checksum so the error detection on the receiving end asks for the packet again.

Connection termination :

Connection will be terminated in a similar way to tcp where the sender will announce the connection termination with a FIN flag and will wait for receiver to respond with ack .